Firing Room Remote Application Software Development

Kan Liu[1]
NASA Kennedy Space Center, Kennedy Space Center, FL 32899

The Engineering and Technology Directorate (NE) at National Aeronautics and Space Administration (NASA) Kennedy Space Center (KSC) is designing a new command and control system for the checkout and launch of Space Launch System (SLS) and future rockets. The purposes of the semester long internship as a remote application software developer include the design, development, integration, and verification of the software and hardware in the firing rooms, in particular with the Mobile Launcher (ML) Launch Accessories (LACC) subsystem. In addition, a software test verification procedure document was created to verify and checkout LACC software for Launch Equipment Test Facility (LETF) testing.

Nomenclature

ACL	=	Application Control Language
ASPU	=	Aft Skirt Purge Umbilical
CAA	=	Crew Access Arm
CSFSU	=	Core Stage Forward Skirt Umbilical
CSITU	=	Core Stage InterTank Umbilical
EDL	=	Engineering Development Laboratory
GN2	=	Gaseous Nitrogen
HAASP	=	Hydraulic Arms and Accessories Service Pressure
ICPSU	=	Interim Cryogenic Propulsive Stage Umbilical
ILOA	=	Integrated Launch Operations and Applications
KSC	=	Kennedy Space Center
LCS	=	Launch Control System
LCC	=	Launch Control Center
LETF	=	Launch Equipment Test Facility
ML	=	Mobile Launcher
NASA	=	National Aeronautics and Space Administration
NE	=	Engineering and Technology Directorate
OSMU	=	Orion Service Module Umbilical
SLS	=	Space Launch System
TSMU	=	Tail Service Mast Umbilical
VM	=	Virtual Machine
VS	=	Vertical Stabilizer

I. Introduction

In order to check out and launch rockets at National Aeronautics and Space Administration (NASA) Kennedy Space Center (KSC), an abundance of hardware components and parts are used and grouped into multiple subsystems. A new command and control system was needed to support the Space Launch System (SLS) as it had been undergoing heavy development in the past several years and was expected to be fully working in 2018. As a remote application software developer, a series of tasks are required, including getting trained, design, development, integration, and verification on the software and hardware for the firing rooms. Remote displays use Java technology on a Linux platform while the display development consists of drag and drop. An abstraction layer called the Application Control

[1] Undergraduate Student, Dept. of Aeronautics and Astronautics, The Ohio State University

Language (ACL) sits on top of the C++ language and allows non-software engineers to comprehend the source code for the individual subsystem, which could potentially be maintained for 30 years or more.

The initial phase of the project included getting familiar with the skills and abilities required to create a remote control application and a remote display using the tools on the Linux Virtual Machine (VM). After the completion of the initial training and verification by a mentor, the Launch Accessories hydraulic umbilical arm subsystem was assigned to me. As a remote display developer, I had to follow the particular team's schedule and project goals. The software development process of designing and documenting the software, developing the suitable software, testing and documenting the unit, integrating the software with launch control system hardware and models, fixing and documenting bug fixes, writing and executing verification tests were carried out in sequence.

The duration of the internship did not me to complete a full software development life cycle. But it was crucial to complete and perform the work required for the particular phase of the specific subsystem team.

II. Objectives

The overall goal of the fall internship was to assist Launch Accessories subsystem team. In order to meet the goal and requirements, the following objectives were set and to be completed in sequence:

1. Since I am a returning intern and have already been trained, only basic review on the wiki website and re-familiar with the tools and skills were required. 3. Hydraulic umbilical arms subsystem team was assigned. This particular subsystem is currently undergoing level V software testing for local displays. Launch Accessories (LACC) Verification Test Procedure is to be written and modified based on needs.

4. After completion of level V software testing in the firing room of Launch Control Center (LCC), the next phase of testing is carried out in the Launch Equipment Test Facility (LETF). This phase is called LETF testing designed to integrate umbilical arms hardware and software in a safe and secure setting.

5. The remote developer is also responsible to create hydraulic umbilical arm displays, to modify the displays based on customer needs and requirements, to fix any bugs or issues, and to integrate the displays with the simulation model and hardware.

6. To provide proof that the subsystem team or the software lead can access the software and documentation along with proof that the software lead is satisfied with the produced software products. Also to make sure that the new subsystem team members are familiar with the integration process, are comfortable with the development sets, and running the verification procedure.

III. Setup

The design and development part of the process was completed in the offices of the KSC Operation Support Building II with the support of mentors and experienced launch command and control personnel. Windows computers and appropriate software were installed with instructions. The software testing and integration part of the project were carried out in the firing rooms of the KSC Launch Control Center (LCC) building with appropriate setup and support provided by the firing room support personnel. Hardware testing and integration were conducted in the Launch Equipment Testing Facility (LETF) of KSC where simulated rocket force and motion were provided and the umbilical arms were installed on a lower platform instead of directly on the Mobile Launcher (ML). The hardware testing and integration, which we called LETF testing, provides a safer and easier testing environment prior to the testing on the ML.

IV. Results

A. Training

The training part of the internship was to be conducted before going out there assisting subsystem teams in actual projects. This was done so that the developer will be familiar with the LCS software and development tools that were being used. Since this is the developer's second rotation of internship, the training part only requires reviewing guides and re-familiarizing of the development tools.

Since I am a returning remote developer to the same subsystem, the training consists of basic refresher courses including reviewing Hitchhiker's Guide, renew IT accounts that are expired and obtain necessary license for that particular software. Skills refresher trainings are then completed by watching all the videos and self-directed training documents on the wiki website. Development software tools were re-visited. The training part was smooth and less time consuming since the developer has already gone through the process.

B. Launch Accessories Subsystem

Figure 1. ML and SLS. Image source: NASAspaceflight[3]

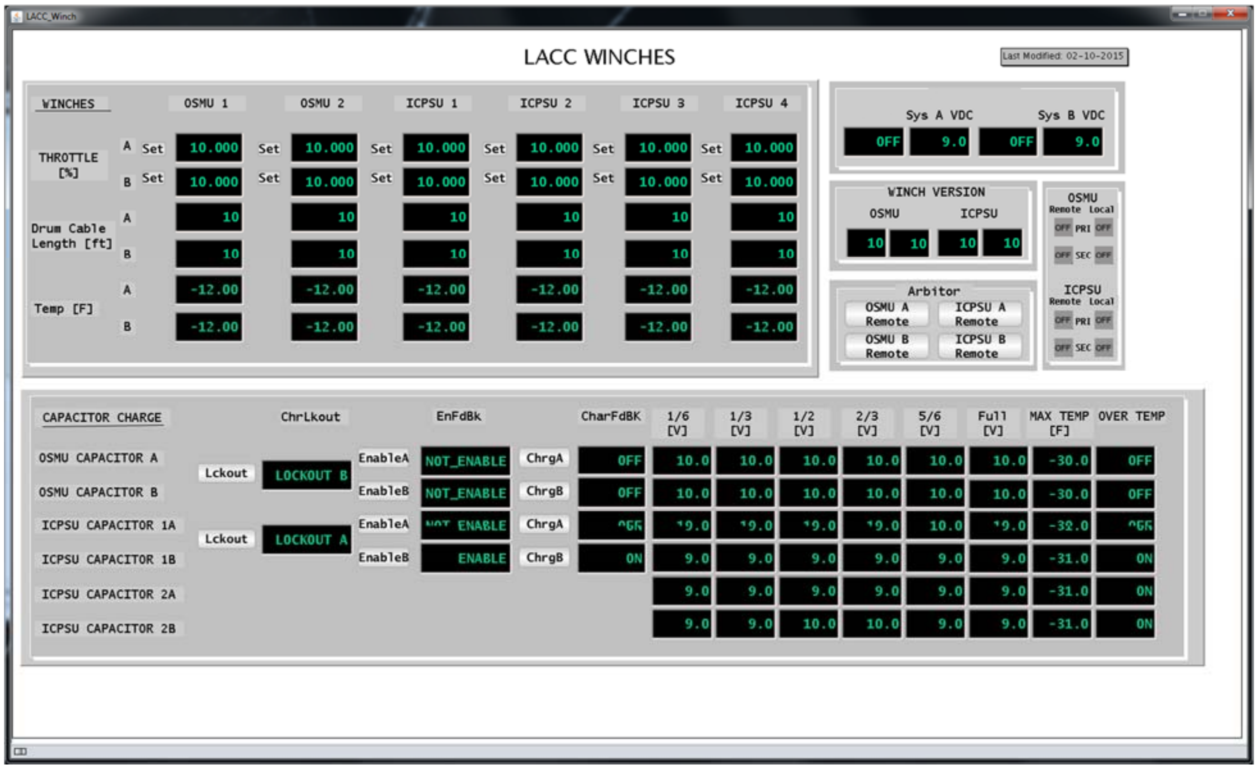
The Launch Accessories Subsystem is made up of fourteen arms and umbilical subsystem. As shown in Figure 1, the SLS will be placed on top of the ML platform. The umbilical arms, which provides crew access, cryogenic, and fuels for SLS, will be installed on the ML and linked to the SLS. The arms monitor the health status and stabilize the vehicle[3]. These arms consist of Crew Access Arm (CAA), Orion Service Module Umbilical (OSMU), Interim Cryogenic Propulsive Stage Umbilical (ICPSU), Core Stage Forward Skirt Umbilical (CSFSU), Core Stage Inter-Tank Umbilical (CSITU), Vehicle Stabilizer (VS), Tail Service Mast Umbilical (TSMU), and Aft Skirt Purge Umbilical (ASPU).

The subsystem had accessories including Hydraulic Arms and Accessories Service Pressure (HAASP), Winch for OSMU and ICPSU, and Gaseous Nitrogen (GN2) Pressure. The objectives of the remote software developers were to make the arms extend to and retract from the vehicle by controlling the valves and hydraulic supply on the remote display. The arm movement is tied directly to the amount of hydraulic pressure supplied. All the umbilical arms are T-0 arms except for CAA, which means automated retract sequences are executed when the rocket launches at T-0. There are latch back mechanisms designed for most of the umbilical arms to prevent damaging the vehicle and the arms themselves. Redundancy in valves and control played an important role in order to carry out a safe and successful mission.

1. LACC OSMU ICPSU Software Verification Procedures

The author entered the software-testing phase of the project and was given task to test and modify the current OSMU Displays. For example figure 2 presents the Winch Display after being modified and reorganized. Level V software testing, which is the testing used to verify the subsystem software before using it against the hardware, was conducted in the firing room of LCC to verify the response of each indicator after forcing the values in the simulation model. Prior to the level V testing, all the global identifiers were checked and verified between the user-generated spreadsheet and the identifier bank in the test configuration. The displays were then tested by forcing values to the global identifiers and observing appropriate changes on the display. Level V testing can be passed only if all the global identifiers on the display are verified to be working properly.

For Level V software testing, a test procedure document was created to ensure each step is executed per software requirements, passed with quality assurance (QA) witness, and showed traceability for future re-test. LACC OSMU ICPSU Software Verification Procedures was written, approved by other subsystem engineers, and promoted for release. This particular test procedure is divided into two parts for two stages of testing. First part of the procedure is strictly dealing with OSMU related displays, indicators, and commands. Once the OSMU verification test cases were developed, Test Readiness Review meeting was held for all the members of the subsystem to poll for go signals. Level V software testing was then conducted with the presence of QA verifying each steps being passed or failed. After successfully running the dry-runs and a formal test, any nonconformance items were recorded and subject to be corrected right away. The software can then be generated as a media and transported to LETF for hardware integration testing. The second part of the procedure includes OSMU and ICPSU, same process was applied but this time ICPSU components are included.



LACC_Winch

LACC WINCHES

Last Modified: 02-10-2015

WINCHES		OSMU 1	OSMU 2	ICPSU 1	ICPSU 2	ICPSU 3	ICPSU 4
THROTTLE [%]	A	Set 10.000	Set 10.000	Set 10.000	Set 10.000	Set 10.000	Set 10.000
	B	Set 10.000	Set 10.000	Set 10.000	Set 10.000	Set 10.000	Set 10.000
Drum Cable Length [ft]	A	10	10	10	10	10	10
	B	10	10	10	10	10	10
Temp [F]	A	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00
	B	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00

Sys A VDC		Sys B VDC	
OFF	9.0	OFF	9.0

WINCH VERSION			
OSMU		ICPSU	
10	10	10	10

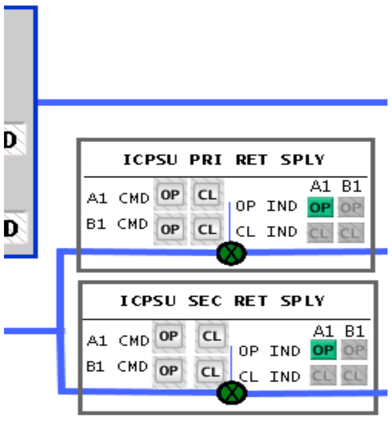
Arbitor: OSMU A Remote, ICPSU A Remote, OSMU B Remote, ICPSU B Remote

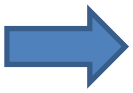
	Remote		Local
OSMU			
	OFF	PRI	OFF
	OFF	SEC	OFF
ICPSU			
	OFF	PRI	OFF
	OFF	SEC	OFF

CAPACITOR CHARGE	ChrLkout		EnFdBk		CharFdBK		1/6 [V]	1/3 [V]	1/2 [V]	2/3 [V]	5/6 [V]	Full [V]	MAX TEMP [F]	OVER TEMP
OSMU CAPACITOR A	Lckout	LOCKOUT B	EnableA	NOT_ENABLE	ChrgA	OFF	10.0	10.0	10.0	10.0	10.0	10.0	-30.0	OFF
OSMU CAPACITOR B			EnableB	NOT_ENABLE	ChrgB	OFF	10.0	10.0	10.0	10.0	10.0	10.0	-30.0	OFF
ICPSU CAPACITOR 1A	Lckout	LOCKOUT A	EnableA	NOT ENABLE	ChrgA	[illegible]	9.0	9.0	9.0	9.0	10.0	9.0	-32.0	[illegible]
ICPSU CAPACITOR 1B			EnableB	ENABLE	ChrgB	ON	9.0	9.0	9.0	9.0	9.0	9.0	-31.0	ON
ICPSU CAPACITOR 2A							9.0	9.0	9.0	9.0	9.0	9.0	-31.0	ON
ICPSU CAPACITOR 2B							9.0	9.0	10.0	10.0	9.0	9.0	-31.0	ON

Figure 2. Winch Remote Display

2. *ICPSU Displays*





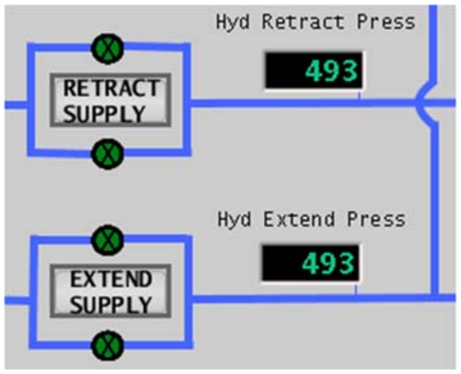


Figure 3. Old ICPSU Display

Figure 4. New ICPSU Display

The previous version of the ICPSU remote display contains everything needed to operate ICPSU. It is important to note that the console users would not be in favor of a complicated display. There was excessive amount of the primary and secondary controls and indicators as shown in Figure 3, which may confuse any first time operator. Design and discussion meetings were attended and the customer demanded a simpler and pop-up style display.

The new ICPSU remote displays simplified and removed most of the valve controls and indicators as shown in Figure 4. If the operator wishes to control or view the primary and secondary indicators of a particular valve, then a pop-up display can be opened by clicking on the display navigation button. For example, the ICPSU Extend Supply information can be opened by clicking the EXTEND SUPPLY button and a pop-up display will appear as show in Figure 5. Extend supply, extend return, retract supply, and retract return have been simplified into four pop-up displays. Figure 6 shows another display for ICPSU HAASP Valve. The latch back valves also had unique displays. This new display method simplified the main display in order for console users to monitor multiple subsystems with the least amount of manpower.

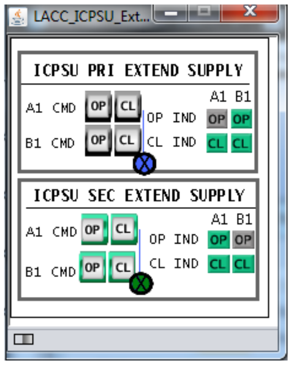


Figure 5. ICPSU HAASP Valve Display

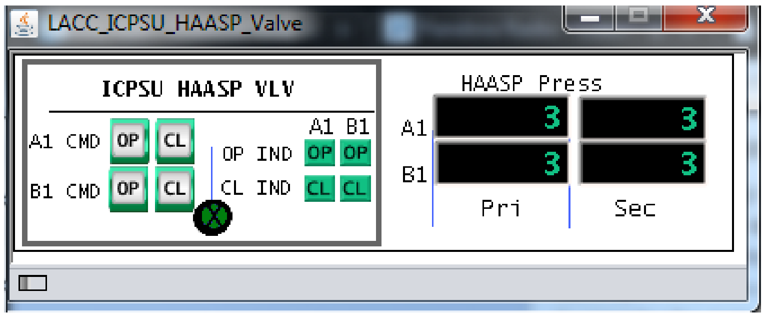


Figure 6. ICPSU Extend Supply Display

3. Power Bus Displays and Engineering Development Lab (EDL)

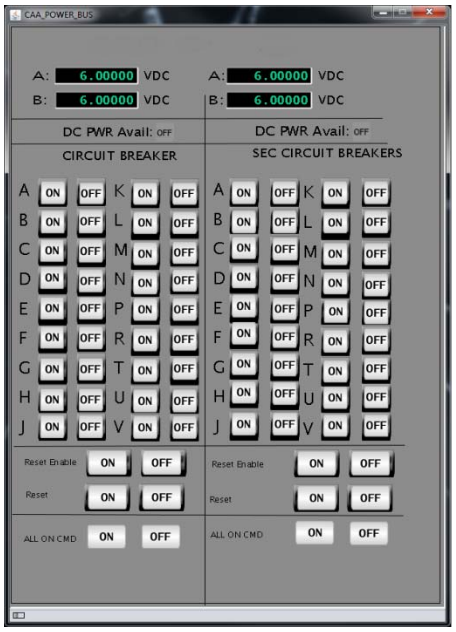


Figure 7. CAA Power Bus

Power Bus Displays were created for CAA, ICPSU, CFSU, and CSITU. The displays were identical in design. The command and indicator identifiers were different depending on the umbilical arm, and the rack numbers were different for each arm.

Figure 7 shows the CAA Power Bus as an example. The rest of the Power Bus displays were originated off of the CAA Power Bus. As shown in the figure, Power Bus consists of circuit breakers for A and B sides. In order for the umbilical arm system to work, all of the breakers must be turned on. This can be done by issuing individual commands to individual breakers until the power available indicator turn on, or All On command can be send to flip all the breakers on.

Other valve and PID displays for CSFSU and CSITU were designed based on ICPSU displays. CSFSU and CSITU were almost identical in terms of arm movement and valve control.

EDL, where most of the local display and logic control development happened, was visited multiple times. Most of the remote displays had to be compatible and similar in design with the local displays. As a remote developer and software test integrator, I had to work and corporate with the developers in EDL. Also physical launch accessories electrical hardware was placed and developed in EDL. For example, the OSMU winch capacitors were in EDL for developers to test.

V. Conclusion

As expected, the duration of the internship did not permit the developer to complete a full software development life cycle, but it was essential to know that documentation is strongly recommended for a multi-phase project. As for deliverables, a modified Winch Display was submitted and integrated to help resolve the current testing issues. The rest of the displays including ICPSU, CSFSU, CSITU, and Power Bus Displays for CAA, ICPSU, CSFSU, and CSITU were created by the developer and are ready in the queue to be promoted for later integration into the system. A level V software verification procedures for OSMU was written and promoted for release version.

Electrical and 90% software requirement document tabletop meetings were attended to discuss the status of the design requirements, and to raise questions and concerns. Also, informal meetings were held with subsystem team engineers to better understand and simplify the umbilical arm displays. An abundance of time was spent in the firing room for level V testing on the Winch Display for OSMU and ICPSU. The basic operation procedure for the firing room integrated command and control system was taught by the firing room experts to launch the displays and to use the tools. The remote developer had been working closely with the local developers in the Engineering Development Laboratory (EDL) to perfect the local Allen Bradley display, especially fixing issues on the Local Winch Display in order for OSMU LETF Testing to be carried out on time. OSMU LETF Test Planning meetings were attended for preparation of OSMU LETF Testing. Software integration meetings were attended every week to plan and support new software deliveries.

VI. Acknowledgments

The author would like to thank and acknowledge Kurt Leucht and Steve Camick for the support and mentoring work throughout the duration of the internship. Thanks to Dave Stewart, Steve Hoyle, Caylyne Shelton, Gerald Stoltenberg, Bryan Yankana, Jolene Hall, Kevin Teufer, and many other KSC professionals for additional mentoring. Also special thanks to KSC Engineering and Technology Directorate and the Education Office, without whom this project would not have been possible.

VII. References

[2]Hitchhiker's Guide to Subsystem Apps and Displays. (2014). [online] ILOA wiki website [Accessed 17 April. 2015].

[3]"NASA Outlines SLS Mobile Launcher Umbilical Plans." NASASpaceFlight.com. N.p., n.d. Web. http://www.nasaspaceflight.com/2012/11/nasa-sls-mobile-launcher-umbilical-plans/ [Accessed 17 April. 2015].

[4]nasa-ice, (2014). LCS Integrated Launch Operations Application Software. [online [Accessed 17 April. 2015].